64-bit Intel® Xeon™ Processor MP with 1 MB L2 Cache

Specification Update

August 2005

Notice: The *64-bit Intel® Xeon™ Processor MP with 1 MB L2 Cache Specification Update* may contain design defects or errors known as errata that may cause the product to deviate from published specifications. Current characterized errata are documented in this specification update.

Document Number: 306752-003

Contents

Revision History

Version	Description	Date
-001	Initial revision of 64-bit Intel® Xeon™ processor MP with 1 MB L2 Cache Specification Update	March 2005
-002	Added erratum J66.	July 2005
-003	Added errata J67.	August 2005

Preface

This document is an update to the specifications contained in the Affected/Related Documents table below. This document is a compilation of device and documentation errata, specification clarifications, and changes. It is intended for hardware system manufacturers and software developers of applications, operating systems, or tools.

This document may also contain information that was not previously published.

Affected/Related Documents

Title	Document #
64-bit Intel® Xeon™ Processor MP with 1MB L2 Cache Datasheet	306751
IA-32 Intel® Architecture Software Developer's Manual, Volume 1: Basic Architecture	253665
IA-32 Intel® Architecture Software Developer's Manual, Volume 2A: Instruction Set Reference, A-M	253666
IA-32 Intel® Architecture Software Developer's Manual, Volume 2B: Instruction Set Reference, N-Z	253667
IA-32 Intel® Architecture Software Developer's Manual, Volume 3: System Programming Guide	253668

Nomenclature

S-Spec Number is a five-digit code used to identify products. Products are differentiated by their unique characteristics, e.g., core speed, L3 cache size, package type, etc. as described in the processor identification information table. Care should be taken to read all notes associated with each S-Spec number.

Errata are design defects or errors. Errata may cause the processor's behavior to deviate from published specifications. Hardware and software designed to be used with any given processor must assume that all errata documented for that processor are present on all devices unless otherwise noted.

Specification Changes are modifications to the current published specifications. These changes will be incorporated in the next release of the specifications.

Specification Clarifications describe a specification in greater detail or further highlight a specification's impact to a complex design situation. These clarifications will be incorporated in the next release of the specifications.

Documentation Changes include typos, errors, or omissions from the current published specifications. These changes will be incorporated in the next release of the specifications.

Summary Table of Changes

The following table indicates the Errata, Documentation Changes, Specification Clarifications, or Specification Changes that apply to Intel processors. Intel intends to fix some of the errata in a future stepping of the component, and to account for the other outstanding issues through documentation or specification changes as noted. This table uses the following notation:

Codes Used in Summary Table

Codes Used in Summary Table

X:	Specification Change, Erratum, Specification Clarification, or Documentation Change applies to the given processor stepping
(No mark) or Blank Box:	This item is fixed in or does not apply to the given stepping.
Doc:	Document change or update that will be implemented in a future revision.
Plan Fix:	This erratum may be fixed in a future stepping of the product.
Fixed:	This erratum has been previously fixed.
No Fix:	There are no plans to fix this erratum.
	Change bar to left of table row indicates this item is either new or modified from the previous version of this document.

Each Specification Update item will be prefixed with a capital letter to distinguish the product. The key below details the letters that are used in Intel's microprocessor Specification Updates:

A = Intel® Pentium® II processor
B = Mobile Intel® Pentium® II processor
C = Intel® Celeron® processor
D = Intel® Pentium® II Xeon™ processor
E = Intel® Pentium® III processor
F = Intel® Pentium® 4 processor Extreme Edition and Intel® Pentium® D processor
G = Intel® Pentium® III Xeon™ processor
H = Mobile Intel® Celeron® processor at 466/433/400/366/333/300 and 266 MHz
J = 64-bit Intel® Xeon™ Processor MP with 1MB L2 Cache
K = Mobile Intel® Pentium® III processor
L = Intel® Celeron® D processor
M = Mobile Intel® Celeron® processor
N = Intel® Pentium® 4 processor
O = Intel® Xeon™ processor MP
P = Intel® Xeon™ processor
Q = Mobile Intel® Pentium® 4 processor supporting Hyper-Threading Technology on 90-nm process technology
R = Intel® Pentium® 4 processor on 90 nm process

S = 64-bit Intel® Xeon™ processor with 800 MHz System Bus (1 MB and 2 MB L2 Cache versions)
T = Mobile Intel® Pentium® 4 processor-M
U = 64-bit Intel® Xeon™ Processor MP with up to 8 MB L3 Cache
V = Mobile Intel® Celeron® processor on .13 Micron Process in Micro-FCPGA Package
W = Intel® Celeron® M processor
X = Intel® Pentium® M processor on 90 nm process with 2-MB L2 Cache
Y = Intel® Pentium® M processor
Z = Mobile Intel® Pentium® 4 processor with 533 MHz system bus

The Specification Updates for the Pentium® processor, Pentium® Pro processor, and other Intel products do not use this convention.

Errata (Sheet 1 of 3)

No.	A-0/ 0F41h	Plans	Errata
J1	X	No Fix	Transaction is not retired after BINIT#
J2	X	No Fix	Invalid opcode 0FFFh requires A ModRM byte
J3	X	No Fix	Processor may hang due to speculative page walks to non-existent system memory
J4	X	No Fix	Memory type of the load lock different from its corresponding store unlock
J5	X	No Fix	Machine check architecture error reporting and recovery may not work as expected
J6	X	No Fix	Debug mechanisms may not function as expected
J7	X	No Fix	Cascading of performance counters does not work correctly when forced overflow is enabled
J8	X	No Fix	EMON event counting of x87 loads may not work as expected
J9	X	No Fix	System bus interrupt messages without data and which receive a hardfailure response may hang the processor
J10	X	No Fix	The processor signals page fault exception (#PF) instead of alignment check exception (#AC) on an unlocked CMPXCHG8B instruction
J11	X	No Fix	FSW may not be completely restored after page fault on FRSTOR or FLDENV instructions
J12	X	No Fix	Processor issues inconsistent transaction size attributes for locked operation
J13	X	No Fix	When the processor is in the system management mode (SMM), debug registers may be fully writeable
J14	X	No Fix	Shutdown and IERR# may result due to a machine check exception on a Hyper-Threading Technology enabled processor
J15	X	No Fix	Processor may hang under certain frequencies and 12.5% STPCLK# duty cycle
J16	X	No Fix	System may hang if a fatal cache error causes Bus Write Line (BWL) transaction to occur to the same cache line address as an outstanding Bus Read Line (BRL) or Bus Read-Invalidate Line (BRIL)
J17	X	No Fix	A write to APIC Task Priority Register (TPR) that lowers priority may seem to have not occurred
J18	X	No Fix	Parity error in the L1 cache may cause the processor to hang
J19	X	No Fix	Sequence of locked operations can cause two threads to receive stale data and cause application hang
J20	X	No Fix	Locks and SMC detection may cause the processor to temporarily hang
J21	X	No Fix	Incorrect Debug Exception (#DB) may occur when a data breakpoint is set on an FP instruction
J22	X	No Fix	xAPIC may not report some illegal vector error
J23	X	No Fix	Enabling No-Eviction Mode (NEM) may prevent the operation of the second logical processor in a Hyper-Threading Technology enabled Boot Strap Processor (BSP)

Errata (Sheet 2 of 3)

No.	A-0/ 0F41h	Plans	Errata
J24	X	No Fix	Task Priority Register (TPR) Updates during voltage transitions of power management events may cause a system hang
J25	X	No Fix	Interactions between the instruction Translation Lookaside Buffer (ITLB) and the instruction streaming buffer may cause unpredictable software behavior
J26	X	No Fix	Incorrect duty cycle is chosen when on-demand clock modulation is enabled in a processor supporting Hyper-Threading Technology
J27	X	No Fix	Memory aliasing of pages as uncacheable memory type and Write Back (WB) may hang the system
J28	X	No Fix	Using STPCLK# and executing code from very slow memory could lead to a system hang
J29	X	No Fix	Processor provides a 4-Byte store unlock after an 8-Byte load lock
J30	X	No Fix	Machine check architecture error reporting and recovery may not work as expected
J31	X	No Fix	Execution of IRET and INTn instructions may cause unexpected system behavior
J32	X	No Fix	Data breakpoints on the high half of a floating-point line split may not be captured
J33	X	No Fix	Machine check exceptions may not update Last-exception Record MSRs (LERs)
J34	X	No Fix	MOV CR3 performs incorrect reserved bit checking when in PAE paging
J35	X	No Fix	Stores to page tables may not be visible to pagewalks for subsequent loads without serializing or invalidating the page table entry
J36	X	No Fix	Recursive page walks may cause a system hang
J37	X	No Fix	VERR/VERW instructions may cause #GP fault when descriptor is in non-canonical space
J38	X	No Fix	The base of a null segment may be non-zero on a processor supporting Intel® Extended Memory 64 Technology (Intel® EM64T)
J39	X	No Fix	Upper 32 Bits of FS/GS with null base may not get cleared in Virtual-8086 Mode on processors with Intel® Extended Memory 64 Technology (Intel® EM64T) Enabled
J40	X	No Fix	Processor may fault when the upper 8 bytes of segment selector is loaded from a far jump through a call gate via the local descriptor table
J41	X	No Fix	Loading a stack segment with a selector that references a non-canonical address can lead to a #SS fault on a processor supporting Intel® Extended Memory 64 Technology (Intel® EM64T)
J42	X	No Fix	FXRSTOR may not restore non-canonical effective addresses on processors with Intel*sup
J43	X	No Fix	A push of ESP that faults may zero the upper 32 bits of RSP
J44	X	No Fix	Enhanced halt state (C1E) voltage transition may affect a system's power management in a Hyper-Threading Technology enabled processor
J45	X	No Fix	Enhanced halt state (C1E) may not be entered in a Hyper-Threading Technology enabled processor
J46	X	No Fix	When the execute disable bit function is enabled a page fault in a mispredicted branch may result in a page fault exception
J47	X	No Fix	Execute disable bit set with AD assist may cause livelock
J48	X	No Fix	The execute disable bit fault may be reported before other types of page fault when both occur
J49	X	No Fix	Writes to IA32_MISC_ENABLE may not update flags for both logical processors
J50	X	No Fix	Execute disable bit set with CR4.PAE may cause livelock
J51	X	No Fix	Checking of page table base address may not match the address bit width supported by the platform
J52	X	No Fix	IA32_MCi_STATUS MSR may improperly indicate that additional MCA information may have been captured

Errata (Sheet 3 of 3)

No.	A-0/ 0F41h	Plans	Errata
J53	X	No Fix	With Trap Flag (TF) asserted, FP instruction that triggers an unmasked FP exception may take single step trap before retirement of instruction
J54	X	No Fix	PDE/PTE loads and continuous locked updates to the same cache line may cause a system livelock
J55	X	No Fix	Branch Trace Store (BTS) and Precise Event Based Sampling (PEBS) may update memory outside the BTS/PEBS buffer
J56	X	No Fix	L-bit of the CS and LMA bit of the IA32_EFER register may have an erroneous value for one instruction following a mode transition in a Hyper-Threading Technology enabled processor supporting Intel® Extended Memory 64 Technology (Intel® EM64T)
J57	X	No Fix	Control Register 2 (CR2) can be updated during a REP MOVS/STOS instruction with fast strings enabled
J58	X	No Fix	REP STOS/MOVS instructions with RCX >= 2^32 may cause a system hang
J59	X	No Fix	An REP MOVS or an REP STOS instruction with RCX >= 2^32 may fail to execute to completion or may write to incorrect memory locations on processors supporting Intel® Extended Memory 64 Technology (Intel® EM64T)
J60	X	No Fix	An REP LODSB or an REP LODSD or an REP LODSQ instruction with RCX >= 2^32 may cause a system hang on processors supporting Intel® Extended Memory 64 Technology (Intel® EM64T)
J61	X	No Fix	CPUID instruction returns incorrect brand string
J62	X	No Fix	Data access which spans both canonical and non-canonical address space may hang system
J63	X	No Fix	Running in System Management Mode (SMM) and l1 data cache adaptive mode may cause unexpected system behavior when SMRAM is mapped to cacheable memory
J64	X	No Fix	A#[39:36] always have On Die Termination (ODT) enabled
J65	X	No Fix	A 64-bit value of Linear Instruction Pointer (LIP) may be reported incorrectly in the Branch Trace Store (BTS) memory record or in the Precise Event Based Sampling (PEBS) memory record
J66	X	Plan Fix	It is possible that two specific invalid opcodes may cause unexpected memory accesses
J67	X	No Fix	At Core-to-bus ratios of 16:1 and above Defer Reply transactions with non-zero REQb values, may cause a Front Side Bus stall

Specification Changes

No.	Specification Changes
	None for this revision of the Specification Update.

Specification Clarifications

No.	Specification Clarifications
J1	Specification Clarification with respect to Time Stamp Counter

Documentation Changes

No.	Documentation Changes
	None for this revision of the Specification Update.

Package Markings

64-bit Intel® Xeon™ Processor MP (604-pin FC-mPGA4 package)

Figure 1. Top-Side Processor Marking Example

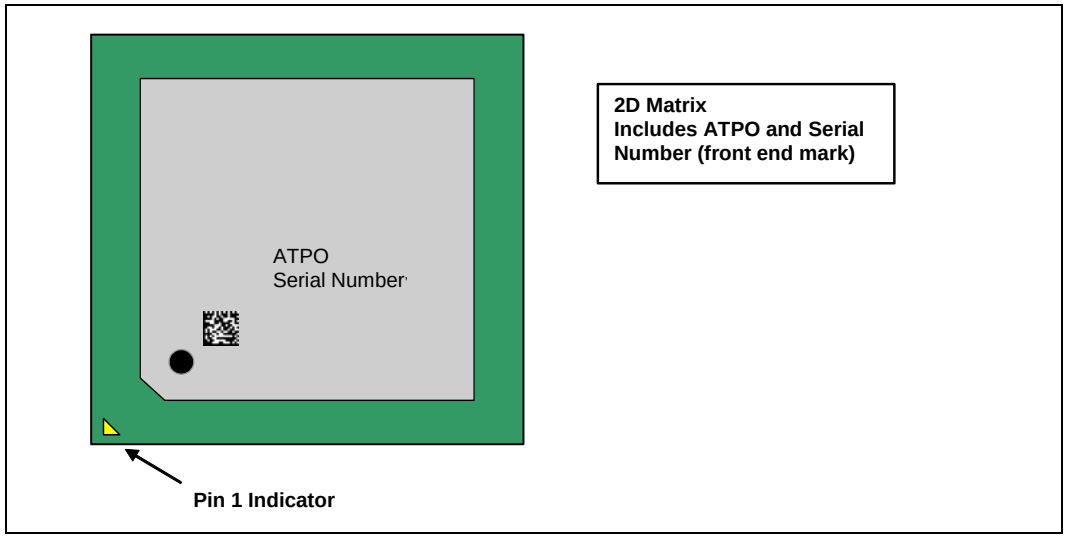
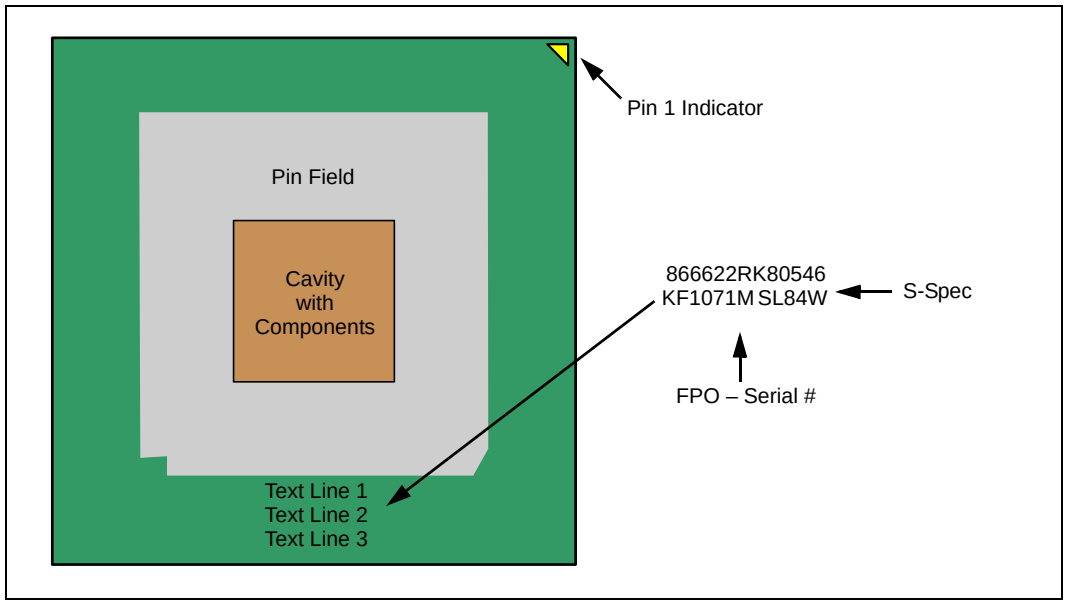


Figure 2. Bottom-Side Processor Marking Example



Identification Information

The 64-bit Intel® Xeon™ processor MP with 1 MB L2 Cache processor can be identified by the following values:

Family[1]	Model[2]
1111b	0100b

NOTES:
1. The Family corresponds to bits [11:8] of the EDX register after RESET, bits [11:8] of the EAX register after the CPUID instruction is executed with a 1 in the EAX register, and the generation field of the Device ID register accessible through Boundary Scan.
2. The Model corresponds to bits [7:4] of the EDX register after RESET, bits [7:4] of the EAX register after the CPUID instruction is executed with a 1 in the EAX register, and the model field of the Device ID register accessible through Boundary Scan.

Cache and TLB descriptor parameters are provided in the EAX, EBX, ECX and EDX registers after the CPUID instruction is executed with a 2 in the EAX register. Please refer to the *AP-485 Intel® Processor Identification and the CPUID Instruction* Application Note for further information on the CPUID instruction.

Identification Information								
64-bit Intel® Xeon™ processor MP with 1 MB L2 Cache processor								
S-Spec	Core Stepping	CPUID	Core Freq (GHz)	Data Bus Freq (MHz)	L2 Cache Size	Processor Package Revision	Package And Revision	Notes
SL84U	A-0	0F41h	3.16	667	1 MB	01	604-pin micro-PGA with 42.5 x 42.5 mm FC-PGA4 package	1,2,4,5
SL84W	A-0	0F41h	3.66	667	1 MB	01		1,2,3,4,5

NOTES:
1. These parts have Tcontrol programmed.
2. These parts are enabled for Intel® Extended Memory 64 Technology.
3. These parts have Thermal Monitor 2 (TM2) feature enabled.
4. These parts are enabled for Enhanced Intel SpeedStep® Technology (EIST).
5. These parts are enabled for Enhanced Halt State (CIE).

Errata

J1. Transaction is not retired after BINIT#

Problem: If the first transaction of a locked sequence receives a HITM# and DEFER# during the snoop phase it should be retried and the locked sequence restarted. However, if BINIT# is also asserted during this transaction, the transaction will not be retried.

Implication: When this erratum occurs, locked transactions will not be retried.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Table of Changes*.

J2. Invalid opcode 0FFFh requires A ModRM byte

Problem: Some invalid opcodes require a ModRM byte and other following bytes, while others do not. The invalid opcode 0FFFh did not require a ModRM in previous generation microprocessors such as Pentium® II or Pentium III processors, but it is required in the Intel® Xeon™ processor.

Implication: The use of an invalid opcode 0FFFh without the ModRM byte may result in a page or limit fault on the Intel Xeon processor. When this erratum occurs, locked transactions will not be retried.

Workaround: To avoid this erratum use ModRM byte with invalid 0FFFh opcode.

Status: For the steppings affected, see the *Summary Table of Changes*.

J3. Processor may hang due to speculative page walks to non-existent system memory

Problem: a load operation issued speculatively by the processor that misses the data translation lookaside buffer (dtlb) results in a page walk. a branch instruction older than the load retires so that this load operation is now in the mispredicted branch path. due to an internal boundary condition, in some instances the load is not canceled before the page walk is issued.

The Page Miss Handler (PMH) starts a speculative page-walk for the Load and issues a cacheable load of the Page Directory Entry (PDE). This PDE load returns data that point to a page table entry in uncacheable (UC) memory. The PMH issues the PTE Load to UC space, which is issued on the Front Side Bus. No response comes back for this load PTE operation since the address is pointing to system memory, which does not exist.

This load to non-existent system memory causes the processor to hang because other bus requests are queued up behind this UC PTE load, which never gets a response. If the load was accessing valid system memory, the speculative page-walk would successfully complete and the processor would continue to make forward progress.

Implication: Processor may hang due to speculative page walks to non-existent system memory.

Workaround: Page directories and page tables in UC memory space must point to system memory that exists.

Status: For the steppings affected, see the *Summary Table of Changes*.

J4. Memory type of the load lock different from its corresponding store unlock

Problem: The Intel Xeon Processor employs a use-once protocol to ensure that a processor in a multiprocessor system may access data that are loaded into its cache on a Read-for-Ownership operation at least once before it is snooped out by another processor. This protocol is necessary to avoid a dual processor livelock scenario where no processor in the system can gain ownership of a line and

modify it before those data are snooped out by another processor. In the case of this erratum, the use-once protocol incorrectly activates for split load lock instructions. A load lock operation accesses data that split across a page boundary with both pages of WB memory type. The use-once protocol activates and the memory type for the split halves get forced to UC. Since use-once does not apply to stores, the store unlock instructions go out as WB memory type. The full sequence on the Bus is: locked partial read (UC), partial read (UC), partial write (WB), locked partial write (WB). The Use-once protocol should not be applied to Load locks.

Implication: When this erratum occurs, the memory type of the load lock will be different than the memory type of the store unlock operation. This behavior (Load Locks and Store Unlocks having different memory types) does not however introduce any functional failures such as system hangs or memory corruption.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Table of Changes*.

J5. Machine check architecture error reporting and recovery may not work as expected

Problem: When the processor detects errors it should attempt to report and/or recover from the error. In the situations described below, the processor does not report and/or recover from the error(s) as intended.

- When a transaction is deferred during the snoop phase and subsequently receives a Hard Failure response, the transaction should be removed from the bus queue so that the processor may proceed. Instead, the transaction is not properly removed from the bus queue, the bus queue is blocked, and the processor will hang.
- When a hardware prefetch results in an uncorrectable tag error in the L2 cache, MC0_STATUS.UNCOR and MC0_STATUS.PCC are set but no Machine Check Exception (MCE) is signaled. No data loss or corruption occurs because the data being prefetched has not been used. If the data location with the uncorrectable tag error is subsequently accessed, an MCE will occur. However, upon this MCE, or any other subsequent MCE, the information for that error will not be logged because MC0_STATUS.UNCOR has already been set and the MCA status registers will not contain information about the error which caused the MCE assertion but instead will contain information about the prefetch error event.
- When the reporting of errors is disabled for Machine Check Architecture (MCA) Bank 2 by setting all MC2_CTL register bits to 0, uncorrectable errors should be logged in the IA32_MC2_STATUS register but no machine-check exception should be generated. Uncorrectable loads on bank 2, which would normally be logged in the IA32_MC2_STATUS register, are not logged.
- When one half of a 64 byte instruction fetch from the L2 cache has an uncorrectable error and the other 32 byte half of the same fetch from the L2 cache has a correctable error, the processor will attempt to correct the correctable error but cannot proceed due to the uncorrectable error. When this occurs the processor will hang.
- When an L1 cache parity error occurs, the cache controller logic should write the physical address of the data memory location that produced that error into the IA32_MC1_ADDR REGISTER (MC1_ADDR). In some instances of a parity error on a load operation that hits the L1 cache, however, the cache controller logic may write the physical address from a subsequent load or store operation into the IA32_MC1_ADDR register.
- When an error exists in the tag field of a cache line such that a request for ownership (RFO) issued by the processor hits multiple tag fields in the L2 cache (the correct tag and the tag with the error) and the accessed data information also has a correctable error, the processor will correctly log the multiple tag match error but will hang when attempting to execute the machine check exception handler.

- If a memory access receives a machine check error on both 64 byte halves of a 128-byte L2 cache sector, the IA32_MC0_STATUS register records this event as multiple errors, i.e., the valid error bit and the overflow error bit are both set indicating that a machine check error occurred while the results of a previous error were in the error-reporting bank. The IA32_MC1_STATUS register should also record this event as multiple errors but instead records this event as only one correctable error.
- The overflow bit should be set to indicate when more than one error has occurred. The overflow bit being set indicates that more than one error has occurred. Because of this erratum, if any further errors occur, the MCA overflow bit will not be updated, thereby incorrectly indicating only one error has been received.
- If an I/O instruction (IN, INS, REP INS, OUT, OUTS, or REP OUTS) is being executed, and if the data for this instruction become corrupted, the processor will signal a Machine Check Exception (MCE). If the instruction is directed at a device that is powered down, the processor may also receive an assertion of SMI#. Since MCEs have higher priority, the processor will call the MCE handler, and the SMI# assertion will remain pending. However, while attempting to execute the first instruction of the MCE handler, the SMI# will be recognized and the processor will attempt to execute the SMM handler. If the SMM handler is successfully completed, it will attempt to restart the I/O instruction, but will not have the correct machine state due to the call to the MCE handler. This can lead to failure of the restart and shutdown of the processor.
- If PWRGOOD is de-asserted during a RESET# assertion causing internal glitches, the MCA registers may latch invalid information.
- If RESET# is asserted, then de-asserted, and reasserted, before the processor has cleared the MCA registers, then the information in the MCA registers may not be reliable, regardless of the state or state transitions of PWRGOOD.
- If MCERR# is asserted by one processor and observed by another processor, the observing processor does not log the assertion of MCERR#. The Machine Check Exception (MCE) handler called upon assertion of MCERR# will not have any way to determine the cause of the MCE.
- The Overflow Error bit (bit 62) in the IA32_MC0_STATUS register indicates, when set, that a machine check error occurred while the results of a previous error were still in the error reporting bank (i.e. The Valid bit was set when the new error occurred). If an uncorrectable error is logged in the error-reporting bank and another error occurs, the overflow bit will not be set.
- The MCA Error Code field of the IA32_MC0_STATUS register gets written by a different mechanism than the rest of the register. For uncorrectable errors, the other fields in the IA32_MC0_STATUS register are only updated by the first error. Any further errors that are detected will update the MCA Error Code field without updating the rest of the register, thereby leaving the IA32_MC0_STATUS register with stale information.
- When a speculative load operation hits the L2 cache and receives a correctable error, the IA32_MC1_Status Register may be updated with incorrect information. The IA32_MC1_Status Register should not be updated for speculative loads.
- The processor should only log the address for L1 parity errors in the IA32_MC1_Status register if a valid address is available. If a valid address is not available, the Address Valid bit in the IA32_MC1_Status register should not be set. In instances where an L1 parity error occurs and the address is not available because the linear to physical address translation is not complete or an internal resource conflict has occurred, the Address Valid bit is incorrectly set.
- The processor may hang when an instruction code fetch receives a hard failure response from the Front Side Bus. This occurs because the bus control logic does not return data to the core, leaving the processor empty. IA32_MC0_STATUS MSR does indicate that a hard fail response occurred.

The processor may hang when the following events occur and the machine check exception is enabled, CR4.MCE=1. A processor that has it's STPCLK# pin asserted will internally enter the Stop Grant State and finally issue a Stop Grant Acknowledge special cycle to the bus. If an uncorrectable error is generated during the Stop Grant process it is possible for the Stop Grant special cycle to be issued to the bus before the processor vectors to the machine check handler. Once the chipset receives its last Stop Grant special cycle it is allowed to ignore any bus activity from the processors. As a result, processor accesses to the machine check handler may not be acknowledged, resulting in a processor hang.

Implication: The processor is unable to correctly report and/or recover from certain errors

Workaround: None identified.

Status: For the steppings affected, see the *Summary Table of Changes*.

J6. Debug mechanisms may not function as expected

Problem: If the first transaction of a locked sequence receives a HITM# and DEFER# during the snoop phase it should be retried and the locked sequence restarted. However, if BINIT# is also asserted during this transaction, the transaction will not be Certain debug mechanisms may not function as expected on the processor. The cases are as follows:

- When the following conditions occur: 1) An FLD instruction signals a stack overflow or underflow, 2) the FLD instruction splits a page-boundary or a 64 byte cache line boundary, 3) the instruction matches a Debug Register on the high page or cache line respectively, and 4) the FLD has a stack fault and a memory fault on a split access, the processor will only signal the stack fault and the debug exception will not be taken.
- When a data breakpoint is set on the ninth and/or tenth byte(s) of a floating-point store using the Extended Real data type, and an unmasked floating-point exception occurs on the store, the break point will not be captured.
- When any instruction has multiple debug register matches, and any one of those debug registers is enabled in DR7, all of the matches should be reported in DR6 when the processor goes to the debug handler. This is not true during a REP instruction. As an example, during execution of a REP MOVSW instruction the first iteration a load matches DR0 and DR2 and sets DR6 as FFFF0FF5h. On a subsequent iteration of the instruction, a load matches only DR0. The DR6 register is expected to still contain FFFF0FF5h, but the processor will update DR6 to FFFF0FF1h.

A Data breakpoint that is set on a load to uncacheable memory may be ignored due to an internal segment register access conflict. In this case the system will continue to execute instructions, bypassing the intended breakpoint. Avoiding having instructions that access segment descriptor registers e.g. LGDT, LIDT close to the UC load, and avoiding serialized instructions before the UC load will reduce the occurrence of this erratum.

Implication: Certain debug mechanisms do not function as expected on the processor.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Table of Changes*.

J7. Cascading of performance counters does not work correctly when forced overflow is enabled

Problem: The performance counters are organized into pairs. When the CASCADE bit of the Counter Configuration Control Register (CCCR) is set, a counter that overflows will continue to count in the other counter of the pair. The FORCE_OVF bit forces the counters to overflow on every non-zero increment. When the FORCE_OVF bit is set, the counter overflow bit will be set but the counter no longer cascades.

Implication: The performance counters do not cascade when the FORCE_OVF bit is set.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Table of Changes*.

J8. EMON event counting of x87 loads may not work as expected

Problem: If a performance counter is set to count x87 loads and floating-point exceptions are unmasked, the FPU Operand (Data) Pointer (FDP) may become corrupted.

Implication: When this erratum occurs, FPU Operand (Data) Pointer (FDP) may become corrupted.

Workaround: This erratum will not occur with floating-point exceptions masked. If floating-point exceptions are unmasked, then performance counting of x87 loads should be disabled.

Status: For the steppings affected, see the *Summary Table of Changes*.

J9. System bus interrupt messages without data and which receive a hardfailure response may hang the processor

Problem: When a System Bus agent (processor or chipset) issues an interrupt transaction without data onto the System Bus, and the transaction receives a HardFailure response, a potential processor hang can occur. The processor, which generates an inter-processor interrupt (IPI) that receives HardFailure response, will still log the MCA error event cause as HardFailure, even if the APIC causes a hang. Other processors, which are true targets of the IPI, will also hang on hardfailure-without-data, but will not record an MCA HardFailure event as a cause. If a HardFailure response occurs on a System Bus interrupt message with data, the APIC will complete the operation so as not to hang the processor.

Implication: The processor may hang

Workaround: None identified.

Status: For the steppings affected, see the *Summary Table of Changes*.

J10. The processor signals page fault exception (#PF) instead of alignment check exception (#AC) on an unlocked CMPXCHG8B instruction

Problem: If a page fault exception (#PF) and alignment check exception (#AC) both occur for an unlocked CMPXCHG8B instruction, then #PF will be flagged.

Implication: Software that depends on the #AC before the #PF will be affected since #PF is signaled in this case.

Workaround: Remove the software's dependency on #AC having precedence over #PF. Alternately, correct the page fault in the page fault handler and then restart the faulting instruction.

Status: For the steppings affected, see the *Summary Table of Changes*.

J11. FSW may not be completely restored after page fault on FRSTOR or FLDENV instructions

Problem: If the FPU operating environment or FPU state (operating environment and register stack) being loaded by an FLDENV or FRSTOR instruction wraps around a 64-Kbyte or 4-Gbyte boundary and a page fault (#PF) or segment limit fault (#GP or #SS) occurs on the instruction near the wrap boundary, the upper byte of the FPU status word (FSW) might not be restored. If the fault handler does not restart program execution at the faulting instruction, stale data may exist in the FSW.

Implication: When this erratum occurs, stale data will exist in the FSW.

Workaround: Ensure that the FPU operating environment and FPU state do not cross 64-Kbyte or 4-Gbyte boundaries. Alternately, ensure that the page fault handler restarts program execution at the faulting instruction after correcting the paging problem.

Status: For the steppings affected, see the *Summary Table of Changes*.

J12. Processor issues inconsistent transaction size attributes for locked operation

Problem: When the processor is in the Page Address Extension (PAE) mode and detects the need to set the Access and/or Dirty bits in the page directory or page table entries, the processor sends an 8 byte load lock onto the System Bus. A subsequent 8 byte store unlock is expected, but instead a 4 byte store unlock occurs. Correct data are provided since only the lower bytes change, however external logic monitoring the data transfer may be expecting an 8-byte store unlock.

Implication: No known commercially available chipsets are affected by this erratum.

None identified.

Status: For the steppings affected, see the *Summary Table of Changes*.

J13. When the processor is in the system management mode (SMM), debug registers may be fully writeable

Problem: When in System Management Mode (SMM), the processor executes code and stores data in the SMRAM space. When the processor is in this mode and writes are made to DR6 and DR7, the processor should block writes to the reserved bit locations. Due to this erratum, the processor may not block these writes. This may result in invalid data in the reserved bit locations.

Implication: Reserved bit locations within DR6 and DR7 may become invalid.

Workaround: Software may perform a read/modify/write when writing to DR6 and DR7 to ensure that the values in the reserved bits are maintained.

Status: For the steppings affected, see the *Summary Table of Changes*.

J14. Shutdown and IERR# may result due to a machine check exception on a Hyper-Threading Technology enabled processor

Problem: When a Machine Check Exception (MCE) occurs due to an internal error, both logical processors on a Hyper-Threading (HT) Technology enabled processor normally vector to the MCE handler. However, if one of the logical processors is in the “Wait for SIPI” state, that logical processor will not have a MCE handler and will shut down and assert IERR#.

Implication: A processor with a logical processor in the “Wait for SIPI” state will shut down when an MCE occurs on the other thread.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Table of Changes*.

J15. Processor may hang under certain frequencies and 12.5% STPCLK# duty cycle

Problem: If a system de-asserts STPCLK# at a 12.5% duty cycle, and the processor is running below 2 GHz, and the processor thermal control circuit (TCC) on-demand clock modulation is active, the processor may hang. This erratum does not occur under the automatic mode of the TCC.

Implication: When this erratum occurs, the processor will hang.

Workaround: If use of the on-demand mode of the processor's TCC is desired in conjunction with STPCLK# modulation, then assure that STPCLK# is not asserted at a 12.5% duty cycle.

Status: For the steppings affected, see the *Summary Table of Changes*.

J16. System may hang if a fatal cache error causes Bus Write Line (BWL) transaction to occur to the same cache line address as an outstanding Bus Read Line (BRL) or Bus Read-Invalidate Line (BRIL)

Problem: A processor internal cache fatal data ECC error may cause the processor to issue a BWL transaction to the same cache line address as an outstanding BRL or BRIL. As it is not typical behavior for a single processor to have a BWL and a BRL/BRIL concurrently outstanding to the same address, this may represent an unexpected scenario to system logic within the chipset.

Implication: The processor may not be able to fully execute the machine check handler in response to the fatal cache error if system logic does not ensure forward progress on the System Bus under this scenario.

Workaround: System logic should ensure completion of the outstanding transactions. Note that during recovery from a fatal data ECC error, memory image coherency of the BWL with respect to BRL/BRIL transactions is not important. Forward progress is the primary requirement.

Status: For the steppings affected, see the *Summary Table of Changes*.

J17. A write to APIC Task Priority Register (TPR) that lowers priority may seem to have not occurred

Problem: Uncacheable stores to the APIC space are handled in a non-synchronous way with respect to the speed at which instructions are retired. If an instruction that masks the interrupt flag e.g. CLI is executed soon after an uncacheable write to the TPR that lowers the APIC priority the interrupt masking operation may take effect before the actual priority has been lowered. This may cause interrupts whose priority is lower than the initial TPR but higher than the final TPR to not be serviced until the interrupt flag is finally cleared e.g. STI. Interrupts will remain pended and are not lost.

Implication: This condition may allow interrupts to be accepted by the processor but may delay their service

Workaround: This can be avoided by issuing a TPR Read after a TPR Write that lowers the TPR value. This will force the store to the APIC priority resolution logic before any subsequent instructions are executed. No commercial operating system is known to be impacted by this erratum.

Status: For the steppings affected, see the *Summary Table of Changes*.

J18. Parity error in the L1 cache may cause the processor to hang

Problem: If a locked operation accesses a line in the L1 cache that has a parity error, it is possible that the processor may hang while trying to evict the line.

Implication: If this erratum occurs, it may result in a system hang. Intel has not observed this erratum with any commercially available software.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Table of Changes*.

J19. Sequence of locked operations can cause two threads to receive stale data and cause application hang

Problem: While going through a sequence of locked operations, it is possible for the two threads to receive stale data. This is a violation of expected memory ordering rules and causes the application to hang.

Implication: When this erratum occurs in an HT Technology enabled system, an application may hang.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Table of Changes*.

J20. Locks and SMC detection may cause the processor to temporarily hang

Problem: The processor may temporarily hang in an HT Technology enabled system, if one logical processor executes a synchronization loop that includes one or more bus locks and is waiting for release by the other logical processor. If the releasing logical processor is executing instructions that are within the detection range of the self modifying code (SMC) logic, then the processor may be locked in the synchronization loop until the arrival of an interrupt or other event.

Implication: If this erratum occurs in an HT Technology enabled system, the application may temporarily stop making forward progress. Intel has not observed this erratum with any commercially available software.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Table of Changes*.

J21. Incorrect Debug Exception (#DB) may occur when a data breakpoint is set on an FP instruction

Problem: The default Microcode Floating-Point Event Handler routine executes a series of loads to obtain data about the FP instruction that are causing the FP event. If a data breakpoint is set on the instruction causing the FP event, the load in the microcode routine will trigger the data breakpoint resulting in a Debug Exception.

Implication: An incorrect Debug Exception (#DB) may occur if data breakpoint is placed on an FP instruction. Intel has not observed this erratum with any commercially available software or system.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Table of Changes*.

J22. xAPIC may not report some illegal vector error

Problem: The local xAPIC has an Error Status Register, which records all errors it detects. Bit 6 of this register, the Receive Illegal Vector bit, is set when the local xAPIC detects an illegal vector in a message that it receives. When an illegal vector error is received on the same internal clock that the error status register is being written due to a previous error, bit 6 does not get set and illegal vector errors are not flagged.

Implication: The xAPIC may not report some Illegal Vector errors when they occur at approximately the same time as other xAPIC errors. The other xAPIC errors will continue to be reported.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Table of Changes*.

J23. Enabling No-Eviction Mode (NEM) may prevent the operation of the second logical processor in a Hyper-Threading Technology enabled Boot Strap Processor (BSP)

Problem: In an HT Technology enabled system, when NEM is enabled by setting Bit 0 of MSR 080h (IA32_BIOS_CACHE_AS_RAM), the second logical processor associated with the BSP may fail to wake up from “Wait-for-SIPI” state.

Implication: In an HT Technology enabled system, the second logical processor associated with the BSP may not respond to SIPI. The OS will continue to operate but with one less logical processor than expected.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Table of Changes*.

J24. Task Priority Register (TPR) Updates during voltage transitions of power management events may cause a system hang

Problem: Systems with Echo TPR Disable (R/W) bit (bit [23] of IA32_MISC_ENABLE register) set to '0' (default), where xTPR messages are being transmitted on the system bus to the processor, may experience a system hang during voltage transitions caused by the power management events.

Implication: This may cause a system hang during voltage transitions of power management events.

Workaround: It is possible for the BIOS to contain a workaround for this erratum. The BIOS workaround disables the Echo TPR updates on affected steppings.

Status: For the steppings affected, see the *Summary Table of Changes*.

J25. Interactions between the instruction Translation Lookaside Buffer (ITLB) and the instruction streaming buffer may cause unpredictable software behavior

Problem: Complex interactions within the instruction fetch/decode unit may make it possible for the processor to execute instructions from an internal streaming buffer containing stale or incorrect information.

Implication: When this erratum occurs, an incorrect instruction stream may be executed resulting in unpredictable software behavior.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Table of Changes*.

J26. Incorrect duty cycle is chosen when on-demand clock modulation is enabled in a processor supporting Hyper-Threading Technology

Problem: When a processor supporting HT Technology enables On-Demand Clock Modulation on both logical processors, the processor is expected to select the lowest duty cycle of the two potentially different values. When one logical processor enters the AUTOHALT state, the duty cycle implemented should be unaffected by the halted logical processor. Due to this erratum, the duty cycle is incorrectly chosen to be the higher duty cycle of both logical processors.

Implication: Due to this erratum, higher duty cycle may be chosen when the On-Demand Clock Modulation is enabled on both logical processors.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Table of Changes*.

J27. Memory aliasing of pages as uncacheable memory type and Write Back (WB) may hang the system

Problem: When a page is being accessed as either Uncacheable (UC) or Write Combining (WC) and WB, under certain bus and memory timing conditions, the system may loop in a continual sequence of UC fetch, implicit writeback, and Request For Ownership (RFO) retries

Implication: This erratum has not been observed in any commercially available operating system or application. The aliasing of memory regions, a condition necessary for this erratum to occur, is documented as being unsupported in the *IA-32 Intel® Architecture Software Developer's Manual*, Volume 3, section 10.12.4, Programming the PAT. However, if this erratum occurs the system may hang

Workaround: The pages should not be mapped as either UC or WC and WB at the same time.

Status: For the steppings affected, see the *Summary Table of Changes*.

J28. Using STPCLK# and executing code from very slow memory could lead to a system hang

Problem: The system may hang when the following conditions are met:

1. Periodic STPCLK# mechanism is enabled via the chipset
2. Hyper-Threading Technology is enabled
3. One logical processor is waiting for an event (i.e. hardware interrupt)
4. The other logical processor executes code from very slow memory such that every code fetch is deferred long enough for the STPCLK# to be re-asserted.

Implication: If this erratum occurs, the processor will go into and out of the sleep state without making forward progress, since the logical processor will not be able to service any pending event. This erratum has not been observed in any commercial platform running commercial software.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Table of Changes*.

J29. Processor provides a 4-Byte store unlock after an 8-Byte load lock

Problem: When the processor is in the Page Address Extension (PAE) mode and detects the need to set the Access and/or Dirty bits in the page directory or page table entries, the processor sends an 8 byte load lock onto the system bus. A subsequent 8 byte store unlock is expected, but instead a 4 byte store unlock occurs. Correct data information is provided since only the lower bytes change, however external logic monitoring the data transfer may be expecting an 8 byte load lock.

Implication: No known commercially available chipsets are affected by this erratum.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Table of Changes*.

J30. Machine check architecture error reporting and recovery may not work as expected

Problem: When the processor detects errors it should attempt to report and/or recover from the error. In the situations described below, the processor does not report and/or recover from the error(s) as intended.

When a transaction is deferred during the snoop phase and subsequently receives a Hard Failure response, the transaction should be removed from the bus queue so that the processor may proceed. Instead, the transaction is not properly removed from the bus queue, the bus queue is blocked, and the processor will hang.

When a hardware prefetch results in an uncorrectable tag error in the L2 cache, MC0_STATUS.UNCOR and MC0_STATUS.PCC are set but no Machine Check Exception (MCE) is signaled. No data loss or corruption occurs because the data being prefetched have not been used. If the data location with the uncorrectable tag error is subsequently accessed, an MCE will occur. However, upon this MCE, or any other subsequent MCE, the information for that error will not be logged because MC0_STATUS.UNCOR has already been set and the MCA status registers will not contain information about the error which caused the MCE assertion, but instead will contain information about the prefetch error event.

When the reporting of errors is disabled for Machine Check Architecture (MCA) Bank 2 by setting all MC2_CTL register bits to 0, uncorrectable errors should be logged in the IA32_MC2_STATUS register but no machine-check exception should be generated. Uncorrectable loads on bank 2, which would normally be logged in the IA32_MC2_STATUS register, are not logged.

When one-half of a 64-byte instruction fetch from the L2 cache has an uncorrectable error and the other 32-byte half of the same fetch from the L2 cache has a correctable error, the processor will attempt to correct the correctable error but cannot proceed due to the uncorrectable error. When this occurs the processor will hang.

When an L1 cache parity error occurs, the cache controller logic should write the physical address of the data memory location that produced that error into the IA32_MC1_ADDR REGISTER (MC1_ADDR). In some instances of a parity error on a load operation that hits the L1 cache, the cache controller logic may write the physical address from a subsequent load or store operation into the IA32_MC1_ADDR register.

When an error exists in the tag field of a cache line such that a request for ownership (RFO) issued by the processor hits multiple tag fields in the L2 cache (the correct tag and the tag with the error) and the accessed data also have a correctable error, the processor will correctly log the multiple tag match error but will hang when attempting to execute the machine check exception handler.

If a memory access receives a machine check error on both 64 byte halves of a 128-byte L2 cache sector, the IA32_MC0_STATUS register records this event as multiple errors, i.e., the valid error bit and the overflow error bit are both set indicating that a machine check error occurred while the results of a previous error were in the error-reporting bank. The IA32_MC1_STATUS register should also record this event as multiple errors but instead records this event as only one correctable error.

The overflow bit should be set to indicate when more than one error has occurred. The overflow bit being set indicates that more than one error has occurred. Because of this erratum, if any further errors occur, the MCA overflow bit will not be updated, thereby incorrectly indicating only one error has been received.

If an I/O instruction (IN, INS, REP INS, OUT, OUTS, or REP OUTS) is being executed, and if the data for this instruction become corrupted, the processor will signal a Machine Check Exception (MCE). If the instruction is directed at a device that is powered down, the processor may also receive an assertion of SMI#. Since MCEs have higher priority, the processor will call the MCE handler, and the SMI# assertion will remain pending. However, while attempting to execute the first instruction of the MCE handler, the SMI# will be recognized and the processor will attempt to execute the SMM handler. If the SMM handler is successfully completed, it will attempt to restart the I/O instruction, but will not have the correct machine state due to the call to the MCE handler. This can lead to failure of the restart and shutdown of the processor.

If PWRGOOD is de-asserted during a RESET# assertion causing internal glitches, the MCA registers may latch invalid information.

If RESET# is asserted, then de-asserted, and reasserted, before the processor has cleared the MCA registers, then the information in the MCA registers may not be reliable, regardless of the state or state transitions of PWRGOOD.

If MCERR# is asserted by one processor and observed by another processor, the observing processor does not log the assertion of MCERR#. The Machine Check Exception (MCE) handler called upon assertion of MCERR# will not have any way to determine the cause of the MCE.

The Overflow Error bit (bit 62) in the IA32_MC0_STATUS register indicates, when set, that a machine check error occurred while the results of a previous error were still in the error reporting bank (i.e. The Valid bit was set when the new error occurred). If an uncorrectable error is logged in the error-reporting bank and another error occurs, the overflow bit will not be set.

The MCA Error Code field of the IA32_MC0_STATUS register gets written by a different mechanism than the rest of the register. For uncorrectable errors, the other fields in the IA32_MC0_STATUS register are only updated by the first error. Any further errors that are detected will update the MCA Error Code field without updating the rest of the register, thereby leaving the IA32_MC0_STATUS register with stale information.

When a speculative load operation hits the L2 cache and receives a correctable error, the IA32_MC1_Status Register may be updated with incorrect information. The IA32_MC1_Status Register should not be updated for speculative loads.

The processor should only log the address for L1 parity errors in the IA32_MC1_Status register if a valid address is available. If a valid address is not available, the Address Valid bit in the IA32_MC1_Status register should not be set. In instances where an L1 parity error occurs and the address is not available because the linear to physical address translation is not complete or an internal resource conflict has occurred, the Address Valid bit is incorrectly set.

The processor may hang when an instruction code fetch receives a hard failure response from the system bus. This occurs because the bus control logic does not return data to the core, leaving the processor empty. IA32_MC0_STATUS MSR does indicate that a hard fail response occurred.

The processor may hang when the following events occur and the machine check exception is enabled, CR4.MCE=1. A processor that has its STPCLK# pin asserted will internally enter the Stop Grant State and finally issue a Stop Grant Acknowledge special cycle to the bus. If an uncorrectable error is generated during the Stop Grant process it is possible for the Stop Grant special cycle to be issued to the bus before the processor vectors to the machine check handler. Once the chipset receives its last Stop Grant special cycle it is allowed to ignore any bus activity from the processors. As a result, processor accesses to the machine check handler may not be acknowledged, resulting in a processor hang.

Implication: The processor is unable to correctly report and/or recover from certain errors.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Table of Changes*.

J31. Execution of IRET and INTn instructions may cause unexpected system behavior

Problem: There is a small window of time, requiring alignment of many internal microarchitectural events, during which the speculative execution of the IRET or INTn instructions in protected or IA-32e mode may result in unexpected software or system behavior.

Implication: This erratum may result in unexpected instruction execution, events, interrupts or a system hang when the IRET instruction is executed. The execution of the INTn instruction may cause debug breakpoints to be missed.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Table of Changes*.

J32. Data breakpoints on the high half of a floating-point line split may not be captured

Problem: When a floating-point load which splits a 64-byte cache line gets a floating-point stack fault, and a data breakpoint register maps to the high line of the floating-point load, internal boundary conditions exist that may prevent the data breakpoint from being captured.

Implication: When this erratum occurs, a data breakpoint will not be captured.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Table of Changes*.

J33. Machine check exceptions may not update Last-exception Record MSRs (LERs)

Problem: The Last-Exception Record MSRs (LERs) may not get updated when Machine Check Exceptions occur.

Implication: When this erratum occurs, the LER may not contain information relating to the machine check exception. They will contain information relating to the exception prior to the machine check exception.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Table of Changes*.

J34. MOV CR3 performs incorrect reserved bit checking when in PAE paging

Problem: The MOV CR3 instruction should perform reserved bit checking on the upper unimplemented address bits. This checking range should match the address width reported by CPUID instruction 0x8000008. This erratum applies whenever PAE is enabled.

Implication: Software that sets the upper address bits on a MOV CR3 instruction and expects a fault may fail. This erratum has not been observed with commercially available software.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Table of Changes*.

J35. Stores to page tables may not be visible to pagewalks for subsequent loads without serializing or invalidating the page table entry

Problem: Under rare timing circumstances, a page table load on behalf of a programmatically younger memory access may not get data from a programmatically older store to the page table entry if there is not a fencing operation or page translation invalidate operation between the store and the younger memory access. Refer to the *IA-32 Intel® Architecture Software Developer's Manual* for the correct way to update page tables. Software that conforms to the Software Developer's Manual will operate correctly.

Implication: If the guidelines in the *IA-32 Intel® Architecture Software Developer's Manual* are not followed, stale data may be loaded into the processor's Translation Lookaside Buffer (TLB) and used for memory operations. This erratum has not been observed with any commercially available software.

Workaround: The guidelines in the *IA-32 Intel® Architecture Software Developer's Manual* should be followed.

Status: For the steppings affected, see the *Summary Table of Changes*.

J36. Recursive page walks may cause a system hang

Problem: A page walk, accessing the same page table entry multiple times but at different levels of the page table, which causes the page table entry to have its Access bit set may result in a system hang.

Implication: When this erratum occurs, the system may experience a hang.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Table of Changes*.

J37. VERR/VERW instructions may cause #GP fault when descriptor is in non-canonical space

Problem: If a descriptor referenced by the selector specified for the VERR or VERW instructions is in non-canonical space, it may incorrectly cause a #GP fault on a processor supporting Intel EM64T.

Implication: Operating systems or drivers that reference a selector in non-canonical space may experience an unexpected #GP fault. Intel has not observed this erratum with any commercially available software.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Table of Changes*.

J38. The base of a null segment may be non-zero on a processor supporting Intel® Extended Memory 64 Technology (Intel® EM64T)

Problem: In IA-32e mode of the Intel EM64T processor, the base of a null segment may be non-zero.

Implication: Due to this erratum, Intel EM64T enabled systems may encounter unexpected behavior when accessing memory using the null selector.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Table of Changes*.

J39. Upper 32 Bits of FS/GS with null base may not get cleared in Virtual-8086 Mode on processors with Intel® Extended Memory 64 Technology (Intel® EM64T) Enabled

Problem: For processors with Intel EM64T enabled, the upper 32 bits of the FS and GS data segment registers corresponding to a null base may not get cleared when segments are loaded in Virtual-8086 mode.

Implication: This erratum may cause incorrect data to be loaded or stored to memory if FS/GS is not initialized before use in 64-bit mode. Intel has not observed this erratum with any commercially available software.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Table of Changes*.

J40. Processor may fault when the upper 8 bytes of segment selector is loaded from a far jump through a call gate via the local descriptor table

Problem: In IA-32e mode of the Intel EM64T processor, control transfers through a call gate via the Local Descriptor Table (LDT) that uses a 16-byte descriptor, the upper 8-byte access may wrap and access an incorrect descriptor in the LDT. This only occurs on an LDT with a LIMIT>0x10008 with a 16-byte descriptor that has a selector of 0xFFFC.

Implication: In the event this erratum occurs, the upper 8-byte access may wrap and access an incorrect descriptor within the LDT, potentially resulting in a fault or system hang. Intel has not observed this erratum with any commercially available software.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Table of Changes*.

J41. Loading a stack segment with a selector that references a non-canonical address can lead to a #SS fault on a processor supporting Intel® Extended Memory 64 Technology (Intel® EM64T)

Problem: When a processor supporting Intel EM64T is in IA-32e mode, loading a stack segment with a selector which references a non-canonical address will result in a #SS fault instead of a #GP fault.

Implication: When this erratum occurs, Intel EM64T enabled systems may encounter unexpected behavior.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Table of Changes*.

J42. FXRSTOR may not restore non-canonical effective addresses on processors with Intel*sup

Problem: Intel EM64T enabled.

Problem: If an x87 data instruction has been executed with a non-canonical effective address, FXSAVE may store that non-canonical FP Data Pointer (FDP) value into the save image. An FXRSTOR instruction executed with 64-bit operand size may signal a General Protection Fault (#GP) if the FDP or FP Instruction Pointer (FIP) is in non-canonical form.

Implication: When this erratum occurs, Intel EM64T enabled systems may encounter an unintended #GP fault.

Workaround: Software should avoid using non-canonical effective addressing in EM64T enabled processors. BIOS can contain a workaround for this erratum removing the unintended #GP fault on FXRSTOR.

Status: For the steppings affected, see the *Summary Table of Changes*.

J43. A push of ESP that faults may zero the upper 32 bits of RSP

Problem: In the event that a push ESP instruction, that faults, is executed in compatibility mode, the processor will incorrectly zero upper 32-bits of RSP.

Implication: A Push of ESP in compatibility mode will zero the upper 32-bits of RSP. Due to this erratum, this instruction fault may change the contents of RSP. This erratum has not been observed in commercially available software.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Table of Changes*.

J44. Enhanced halt state (C1E) voltage transition may affect a system's power management in a Hyper-Threading Technology enabled processor

Problem: In an HT Technology enabled system, the second logical Processor may fail to wake up from "Wait-for-SIPI" state during a C1E voltage transition.

Implication: This erratum may affect a system's entry into the power management mode offered by the C1E event for HT Technology enabled platforms.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Table of Changes*.

J45. Enhanced halt state (C1E) may not be entered in a Hyper-Threading Technology enabled processor

Problem: If the IA32_MISC_ENABLE MSR (0x1A0) C1E enable bit is not set prior to an INIT event on an HT Technology enabled system, the processor will not enter C1E until the next SIPI wakeup event for the second logical processor.

Implication: Due to this erratum, the processor will not enter C1E state.

Workaround: If C1E is supported in the system, the IA32_MISC_ENABLE MSR should be enabled prior to issuing the first SIPI to the second logical processor.

Status: For the steppings affected, see the *Summary Table of Changes*.

J46. When the execute disable bit function is enabled a page fault in a mispredicted branch may result in a page fault exception

Problem: If a page fault in a mispredicted branch occurs in the ITLB, it should not be reported by the processor. However, if the execute disable bit function is enabled (IA32_EFER.NXE = 1) and there is a page fault in a mispredicted branch in the ITLB, a page fault exception may occur.

Implication: When this erratum occurs, a page fault exception may occur.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Table of Changes*.

J47. Execute disable bit set with AD assist may cause livelock

Problem: If Execute Disable Bit is set and the resulting page requires the processor to set the A and/or D bit (Access and/or Dirty bit) in the PTE, then the processor may livelock.

Implication: When this erratum occurs, the processor may livelock resulting in a system hang or operating system failure.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Table of Changes*.

J48. The execute disable bit fault may be reported before other types of page fault when both occur

Problem: If the execute disable bit is enabled and both the execute disable bit fault and page faults occur, the execute disable bit fault will be reported prior to other types of page fault being reported.

Implication: No impact to properly written code since both types of faults will be generated but in the opposite order.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Table of Changes*.

J49. Writes to IA32_MISC_ENABLE may not update flags for both logical processors

Problem: On processors supporting HT Technology with Execute Disable Bit feature, writes to IA32_MISC_ENABLE may only update IA32_EFER.NXE for the current logical processor.

Implication: Due to this erratum, the non-current logical processor may not update its IA32_EFER.NXE bit.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Table of Changes*.

J50. Execute disable bit set with CR4.PAE may cause livelock

Problem: If the execute disable bit of IA32_MISC_Enable is set along with the PAE bit of CR4 (IA32_EFER.NXE & CR4.PAE), the processor may livelock.

Implication: When this erratum occurs, the processor may livelock resulting in a system hang or operating system failure.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Table of Changes*.

J51. Checking of page table base address may not match the address bit width supported by the platform

Problem: If the page table base address, included in the page map level-4 table, page-directory pointer table, page-directory table or page table, exceeds the physical address range supported by the platform (e.g. 36-bit) and it is less than the implemented address range (e.g. 40-bit), the processor does not check if the address is invalid.

Implication: If software sets such invalid physical address in those tables, the processor does not generate a page fault (#PF) upon access to that virtual address, and the access results in an incorrect read or write. If

BIOS provides only valid physical address ranges to the operating system, this erratum will not occur.

Workaround: BIOS must provide valid physical address ranges to the operating system.

Status: For the steppings affected, see the *Summary Table of Changes*.

J52. IA32_MCi_STATUS MSR may improperly indicate that additional MCA information may have been captured

Problem: When a data parity error is detected and the bus queue is busy, the ADDRV and MISCV bits of the IA32_MCi_STATUS register may be asserted even though the contents of the IA32_MCi_ADDR and IA32_MCi_MISC MSRs were not properly captured.

Implication: If this erratum occurs, the MCA information captured in the IA32_MCi_ADDR and IA32_MCi_MISC may not correspond to the reported machine-check error, even though the ADDRV and MISCV are asserted.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Table of Changes*.

J53. With Trap Flag (TF) asserted, FP instruction that triggers an unmasked FP exception may take single step trap before retirement of instruction

Problem: If an FP instruction generates an unmasked exception with the EFLAGS.TF=1, it is possible for external events to occur, including a transition to a lower power state. When resuming from the lower power state, it may be possible to take the single step trap before the execution of the original FP instruction completes.

Implication: A Single Step trap will be taken when not expected.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Table of Changes*.

J54. PDE/PTE loads and continuous locked updates to the same cache line may cause a system livelock

Problem: In a multiprocessor configuration, if one processor is continuously doing locked updates to a cache line that is being accessed by another processor doing a page table walk, the page table walk may not complete.

Implication: Due to this erratum, the system may livelock until some external event interrupts the locked update. Intel has not observed this erratum with any commercially available software.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Table of Changes*.

J55. Branch Trace Store (BTS) and Precise Event Based Sampling (PEBS) may update memory outside the BTS/PEBS buffer

Problem: If the BTS/PEBS buffer is defined such that:

- The difference between BTS/PEBS buffer base and BTS/PEBS absolute maximum is not an integer multiple of the corresponding record sizes
- BTS/PEBS absolute maximum is less than a record size from the end of the virtual address space
- The record that would cross BTS/PEBS absolute maximum will also continue past the end of the virtual address space

A BTS/PEBS record can be written that will wrap at the 4G boundary (IA32) or 2^64 boundary (Intel EM64T mode), and write memory outside of the BTS/PEBS buffer.

Implication: Software that uses BTS/PEBS near the 4G boundary (IA32) or 2^64 boundary (EM64T mode), and defines the buffer such that it does not hold an integer multiple of records can update memory outside the BTS/PEBS buffer.

Workaround: Define BTS/PEBS buffer such that BTS/PEBS absolute maximum minus BTS/PEBS buffer base is integer multiple of the corresponding record sizes as recommended in the IA-32 Intel® Architecture Software Developer's Manual, Volume 3.

Status: For the steppings affected, see the *Summary Table of Changes*.

J56. L-bit of the CS and LMA bit of the IA32_EFER register may have an erroneous value for one instruction following a mode transition in a Hyper-Threading Technology enabled processor supporting Intel® Extended Memory 64 Technology (Intel® EM64T)

Problem: In an Intel EM64T enabled Processor, the L-bit of the Code Segment (CS) descriptor may not update with the correct value in an HT Technology. This may occur in a small window when one logical processor is making a transition from a compatibility mode to a 64-bit mode (or vice-versa) while the other logical processor is being stalled. A similar problem may occur for the observation of the EFER.LMA bit by the decode logic.

Implication: The first instruction following a mode transition may be decoded as if it was still in the previous mode. For example, this may result in an incorrect stack size used for a stack operation, i.e. a write of only 4-bytes and an adjustment to ESP of only 4 in 64-bit mode. The problem can manifest itself on any instruction which may behave differently in 64-bit mode than in compatibility mode.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Table of Changes*.

J57. Control Register 2 (CR2) can be updated during a REP MOVS/STOS instruction with fast strings enabled

Problem: Under limited circumstances while executing a REP MOVS/STOS string instruction, with fast strings enabled, it is possible for the value in CR2 to be changed as a result of an interim paging event, normally invisible to the user. Any higher priority architectural event that arrives and is handled while the interim paging event is occurring may see the modified value of CR2.

Implication: The value in CR2 is correct at the time that an architectural page fault is signaled. Intel has not observed this erratum with any commercially available software.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Table of Changes*.

J58. REP STOS/MOVS instructions with RCX >= 2^32 may cause a system hang

Problem: In IA-32e mode using Intel EM64T-enabled processors, executing a repeating string instruction with the iteration count greater than or equal to 2^32 and a pending event may cause the REP STOS/MOVS instruction to live lock and hang.

Implication: When this erratum occurs, the processor may live lock and result in a system hang. Intel has not observed this erratum with any commercially available software or system.

Workaround: Do not use strings larger than 4 GB.

Status: For the steppings affected, see the *Summary Table of Changes*.

J59. An REP MOVS or an REP STOS instruction with RCX >= 2^32 may fail to execute to completion or may write to incorrect memory locations on processors supporting Intel® Extended Memory 64 Technology (Intel® EM64T)

Problem: In IA-32e mode using Intel EM64T-enabled processors, an REP MOVS or an REP STOS instruction executed with the register RCX >= 2^32, may fail to execute to completion or may write data to incorrect memory locations.

Implication: This erratum may cause an incomplete instruction execution or incorrect data in the memory. Intel has not observed this erratum with any commercially available software or system.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Table of Changes*.

J60. An REP LODSB or an REP LODSD or an REP LODSQ instruction with RCX >= 2^32 may cause a system hang on processors supporting Intel® Extended Memory 64 Technology (Intel® EM64T)

Problem: In IA-32e mode using Intel EM64T-enabled processors, an REP LODSB or an REP LODSD or an REP LODSQ instruction executed with the register RCX >= 2^32 may fail to complete execution causing a system hang. Additionally, there may be no #GP fault due to the non-canonical address in the RSI register.

Implication: This erratum may cause a system hang on Intel EM64T-enabled platforms. Intel has not observed this erratum with any commercially available software or system.

Workaround: It is possible for the BIOS to contain a workaround for this erratum

Status: For the steppings affected, see the *Summary Table of Changes*.

J61. CPUID instruction returns incorrect brand string

Problem: When a CPUID instruction is executed on an Intel Xeon processor MP the system reports Intel Pentium 4 CPU when it should return the Intel Xeon MP brand string.

Implication: When this erratum occurs, the processor will report the incorrect brand string.

Workaround: It is possible for the BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Table of Changes*.

J62. Data access which spans both canonical and non-canonical address space may hang system

Problem: If a data access causes a page split across the canonical to non-canonical address space, the processor may livelock which in turn would cause a system hang.

Implication: When this erratum occurs, the processor may livelock, resulting in a system hang. Intel has not observed this erratum with any commercially available software.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Table of Changes*.

J63. Running in System Management Mode (SMM) and l1 data cache adaptive mode may cause unexpected system behavior when SMRAM is mapped to cacheable memory

Problem: In a HT Technology-enabled system, unexpected system behavior may occur if a change is made to the value of the CR3 result from an Resume from System Management (RSM) instruction while in

L1 data cache adaptive mode (IA32_MISC_ENABLES MSR 0x1a0, bit 24). This behavior will only be visible when SMRAM is mapped into WB/WT cacheable memory on SMM entry and exit.

Implication: This erratum can have multiple failure symptoms, including incorrect data in memory. Intel has not observed this erratum with any commercially available software.

Workaround: Disable L1 data cache adaptive mode by setting the L1 Data Cache Context Mode control bit (bit 24) of the IA32_MISC_ENABLES MSR (0x1a0) to 1.

Status: For the steppings affected, see the *Summary Table of Changes*.

J64. A#[39:36] always have On Die Termination (ODT) enabled

Problem: ODT will be enabled on the end agents and middle agents for signals A#[39:36], resulting in a V_{OL} increase, which may create system level timing problems.

Implication: Signal quality analysis should be reviewed for these signals. Systems which do not use signals A#[39:36] (typically with < 64 GB of addressable memory) are not exposed to this erratum.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Table of Changes*.

J65. A 64-bit value of Linear Instruction Pointer (LIP) may be reported incorrectly in the Branch Trace Store (BTS) memory record or in the Precise Event Based Sampling (PEBS) memory record

Problem: On a processors supporting Intel EM64T,

- If an instruction fetch wraps around the 4G boundary in Compatibility Mode, the 64-bit value of LIP in the BTS memory record will be incorrect (upper 32 bits will be set to 0xFFFFFFFF when they should be 0).
- If a PEBS event occurs on an instruction whose last byte is at memory location FFFFFFFFh, the 64-bit value of LIP in the PEBS record will be incorrect (upper 32 bits will be set to FFFFFFFFh when they should be 0).

Implication: Intel has not observed this erratum on any commercially available software.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Table of Changes*.

J66. It is possible that two specific invalid opcodes may cause unexpected memory accesses

Problem: A processor is expected to respond with an undefined opcode (#UD) fault when executing either opcode 0F 78 or a Grp 6 Opcode with bits 5:3 of the Mod/RM field set to 6, however the processor may respond instead, with a load to an incorrect address.

Implication: This erratum may cause unpredictable system behavior or system hang.

Workaround: It is possible for BIOS to contain a workaround for this erratum.

Status: For the steppings affected, see the *Summary Table of Changes*.

J67. At Core-to-bus ratios of 16:1 and above Defer Reply transactions with non-zero REQb values, may cause a Front Side Bus stall

Problem: Certain processors are likely to hang the Front Side Bus (FSB) if the following conditions are met:

1. A Defer Reply transaction has a REQb[2:0] value of either 010b, 011b, 100b, 110b, or 111b, and
2. The operating bus ratio is 16:1 or higher

When these conditions are met, the processor may incorrectly - and indefinitely - assert a snoop stall for the Defer Reply transaction. Such an event will block further progress on the FSB.

Implication: If this erratum occurs, the system may hang. Intel has not observed this erratum with any commercially available system.

Workaround: None identified.

Status: For the steppings affected, see the *Summary Table of Changes*.

Specification Changes

There are no new Specification Changes for this month.

The Specification Changes listed in this section apply to the following documents:

1. *64-bit Intel® Xeon™ Processor MP with 1MB L2 Cache Datasheet* (Document Number 306751)
2. *IA-32 Intel® Architecture Software Developer's Manual, Volume 1: Basic Architecture* (Document Number 253665)
3. *IA-32 Intel® Architecture Software Developer's Manual, Volume 2A: Instruction Set Reference, A-M* (Document Number 253666)
4. *IA-32 Intel® Architecture Software Developer's Manual, Volume 2B: Instruction Set Reference, N-Z* (Document Number 253667)
5. *IA-32 Intel® Architecture Software Developer's Manual, Volume 3: System Programming Guide* (Document Number 253668)

All Specification Changes will be incorporated into a future version of the appropriate Intel Xeon processor documentation.

Specification Clarifications

There are no new Specification Clarifications for this month.

The Specification Changes listed in this section apply to the following documents:

1. *64-bit Intel® Xeon™ Processor MP with 1MB L2 Cache Datasheet* (Document Number 306751)
2. *IA-32 Intel® Architecture Software Developer's Manual, Volume 1: Basic Architecture* (Document Number 253665)
3. *IA-32 Intel® Architecture Software Developer's Manual, Volume 2A: Instruction Set Reference, A-M* (Document Number 253666)
4. *IA-32 Intel® Architecture Software Developer's Manual, Volume 2B: Instruction Set Reference, N-Z* (Document Number 253667)
5. *IA-32 Intel® Architecture Software Developer's Manual, Volume 3: System Programming Guide* (Document Number 253668)

All Specification Clarifications will be incorporated into a future version of the appropriate Intel Xeon processor documentation.

J1. Specification Clarification with respect to Time Stamp Counter

In the "Debugging and Performance Monitoring" section (Sections 15.8, 15.10.9 and 15.10.9.3) of the IA-32 Intel® Architecture Software Developer's Manual Volume 3: System Programming Guide, the Time Stamp Counter definition has been updated to include support for the future processors. This change will be incorporated in the next revision of the IA-32 Intel® Architecture Software Developer's Manual.

15.8 Time-Stamp Counter

The IA-32 architecture (beginning with the Pentium® processor) defines a time-stamp counter mechanism that can be used to monitor and identify the relative time occurrence of processor events. The counter's architecture includes the following components:

- **TSC flag** — A feature bit that indicates the availability of the time-stamp counter. The counter is available in an IA-32 processor implementation if the function CPUID.1:EDX.TSC[bit 4] = 1.
- **IA32_TIME_STAMP_COUNTER MSR** (called TSC MSR in P6 family and Pentium processors) — The MSR used as the counter.
- **RDTSC instruction** — An instruction used to read the time-stamp counter.
- **TSD flag** — A control register flag is used to enable or disable the time-stamp counter (enabled if CR4.TSD[bit 2] = 1).

The time-stamp counter (as implemented in the P6 family, Pentium, Pentium M, Pentium 4, and Intel® Xeon™ processors) is a 64-bit counter that is set to 0 following a RESET of the processor. Following a RESET, the counter will increment even when the processor is halted by the HLT instruction or the external STPCLK# pin. Note that the assertion of the external DPSLP# pin may cause the time-stamp counter to stop.

Members of the processor families increment the time-stamp counter differently:

- For Pentium M processors (family [06H], models [09H, 0DH]); for Pentium 4 processors, Intel Xeon processors (family [0FH], models [00H, 01H, or 02H]); and for P6 family processors: the time-stamp counter increments with every internal processor clock cycle. The internal processor clock cycle is determined by the current core-clock to bus-clock ratio. Intel SpeedStep® technology transitions may also impact the processor clock.
- For Pentium 4 processors, Intel Xeon processors (family [0FH], models [03H and higher]): the time-stamp counter increments at a constant rate. That rate may be set by the maximum core-clock to bus-clock ratio of the processor or may be set by the frequency at which the processor is booted. The specific processor configuration determines the behavior. Constant TSC behavior ensures that the duration of each clock tick is uniform and supports the use of the TSC as a wall clock timer even if the processor core changes frequency. This is the architectural behavior moving forward.

Note: To determine average processor clock frequency, Intel recommends the use of Performance Monitoring logic to count processor core clocks over the period of time for which the average is required. See Section 15.10.9 and Appendix A in this manual for more information.

The RDTSC instruction reads the time-stamp counter and is guaranteed to return a monotonically increasing unique value whenever executed, except for a 64-bit counter wraparound. Intel guarantees that the time-stamp counter will not wraparound within 10 years after being reset. The period for counter wrap is longer for Pentium 4, Intel Xeon, P6 family, and Pentium processors.

Normally, the RDTSC instruction can be executed by programs and procedures running at any privilege level and in virtual-8086 mode. The TSD flag allows use of this instruction to be restricted to programs and procedures running at privilege level 0. A secure operating system would set the TSD flag during system initialization to disable user access to the time-stamp counter. An operating system that disables user access to the time-stamp counter should emulate the instruction through a user-accessible programming interface.

The RDTSC instruction is not serializing or ordered with other instructions. It does not necessarily wait until all previous instructions have been executed before reading the counter. Similarly, subsequent instructions may begin execution before the RDTSC instruction operation is performed.

The RDMSR and WRMSR instructions read and write the time-stamp counter, treating the time-stamp counter as an ordinary MSR (address 10H). In the Pentium 4, Intel Xeon, and P6 family processors, all 64-bits of the time-stamp counter are read using RDMSR (just as with RDTSC). When WRMSR is used to write the time-stamp counter on processors before family [0FH], models [03H, 04H]: only the low order 32-bits of the time-stamp counter can be written (the high-order 32 bits are cleared to 0). For family [0FH], models [03H, 04H]: all 64 bits are writeable.

15.10.9 Counting Clocks

The count of cycles, also known as clockticks, forms a the basis for measuring how long a program takes to execute. Clockticks are also used as part of efficiency ratios like cycles per instruction (CPI). Processor clocks may stop ticking under circumstances like the following:

- The processor is halted when there is nothing for the CPU to do. For example, the processor may halt to save power while the computer is servicing an I/O request. When Hyper-Threading Technology is enabled, both logical processors must be halted for performance-monitoring counters to be powered down.
- The processor is asleep as a result of being halted or because of a power-management scheme. There are different levels of sleep. In the some deep sleep levels, the time-stamp counter stops counting.

There are three ways to count processor clock cycles to monitor performance. These are:

- **Non-halted clockticks** — Measures clock cycles in which the specified logical processor is not halted and is not in any power-saving state. When Hyper-Threading Technology is enabled, this these ticks can be measured on a per-logical-processor basis.
- **Non-sleep clockticks** — Measures clock cycles in which the specified physical processor is not in a sleep mode or in a power-saving state. These ticks cannot be measured on a logical-processor basis.
- **Time-stamp counter** — Some processor models permit clock cycles to be measured when the physical processor is not in deep sleep (by using the time-stamp counter and the RDTSC instruction). Note that such ticks cannot be measured on a per-logical-processor basis. See Section 10.8 for detail on processor capabilities.

The first two methods use performance counters and can be set up to cause an interrupt upon overflow (for sampling). They may also be useful where it is easier for a tool to read a performance counter than to use a time stamp counter (the timestamp counter is accessed using the RDTSC instruction).

For applications with a significant amount of I/O, there are two ratios of interest:

- **Non-halted CPI** — Non-halted clockticks/instructions retired measures the CPI for phases where the CPU was being used. This ratio can be measured on a logical-processor basis when Hyper-Threading Technology is enabled.
- **Nominal CPI** — Time-stamp counter ticks/instructions retired measures the CPI over the duration of a program, including those periods when the machine halts while waiting for I/O.

15.10.9.3 Incrementing the Time-Stamp Counter

The time-stamp counter increments when the clock signal on the system bus is active and when the sleep pin is not asserted. The counter value can be read with the RDTSC instruction.

The time-stamp counter and the non-sleep clockticks count may not agree in all cases and for all processors. See Section 10.8 for more information on counter operation.

Documentation Changes

Note: Documentation changes for *IA-32 Intel® Architecture Software Developer's Manual*, Volumes 1, 2A, 2B, and 3 will be posted in the separate document *IA-32 Intel® Architecture Software Developer's Manual Documentation Changes*. Follow the link below to become familiar with this file.

http://developer.intel.com/design/pentium4/specupdt/252046.htm

There are no new Documentation Changes for this month.

The Documentation Changes listed in this section apply to the following documents:

1. *64-bit Intel® Xeon™ Processor MP with 1MB L2 Cache Datasheet* (Document Number 306751)

All Documentation Changes will be incorporated into a future version of the appropriate Intel Xeon processor documentation.